LISTING OF THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (previously presented) A system for determining program usage on a computer, the system comprising:

a plurality of executable software programs constituting software products, each of the software products being constituted of one or more load modules, the load modules being stored in at least one memory of the computer;

an operating system of the computer that controls execution in the computer of the software products through the invocation of respective load modules thereof;

a monitor that is periodically triggered to collect load module execution information;

a filtering facility that is effective to filter at least one previously identified program from the load module execution information;

a correlator that correlates the filtered load module execution information with data that associates load module names with corresponding software products and develops a list of software products executed in the computer over the course of a given time period; and

a reporter that outputs data reflecting the use of the software products in the computer in terms of software product names thereof.

2. (original) The system of claim 1, in which the monitor operates by taking periodic snapshots of the then current state of active processes in the computer.

3. (original) The system of claim 2, in which the load module execution information includes one or more of the following data items: module names, user names, the time when processes were started, how much CPU time has been used, and a directory path name from which processes were installed.

4. (original) The system of claim 2, including a facility that allows adjusting the period between snapshots in response to program usage activity levels.

5. (original) The system of claim 2, in which the monitor produces a list of executing load modules and their respective directory path names.

6. (original) The system of claim 2, including a facility that determines how many processes have begun and ended between snapshots.

7. (original) The system of claim 2, including a facility that compares successive snapshots to determine which modules have executed and how many were missed.

Claim 8 (canceled).

9. (original) The system of claim 1, including in the load module execution information module names and other process related information including directory, start time, and process ID.

10. (original) The system of claim 2, in which processes are identified by PID (Process IDs) numbers and the monitor includes a facility that obtains a measure of missed processes by calculating M minus H minus E, where M represents a current PID, H represents the highest PID found in a prior snapshot and E represents a count of all processes that have begun since the prior snapshot and are still executing.

11. (original) The system of claim 1, in which the correlator operates in conjunction with a knowledge base that associates load module names with software product names.

12. (original) The system of claim 1, further including a surveying program that develops an inventory of substantially all software products on the computer and a facility which produces a list of non-used software products based on comparing the inventory of software products against the data outputted by the reporter which reflects the use of the software products in the computer.

13. (original) The system of claim 11, in which the knowledge base is a database of records which also associates file names to software products that use them and additionally includes at least one of the following: flags indicating if a module is used uniquely or shared among vendor products; a number indicating file matches required for correlation with the product; file type; file size; file creation date; and embedded strings of text.

14. (original) The system of claim 1, in which the correlator operates by correlating module usage data with an inventory of software products that itself has been obtained by correlating in a knowledge base load module names with software product names.

15. (previously presented) A system for determining program usage on a computer, the system comprising;

a plurality of executable software programs constituting software products, each software product being constituted in turn of one or more load modules, the load modules being stored in at least one memory of the computer;

an operating system of the computer that controls execution in the computer of the software products through the invocation of respective load modules thereof;

a monitor that collects load module execution information by deducing which load modules are being used in given processes of the computer, without directly monitoring the actual invocation by the operating system of the load modules;

a filtering facility that is effective to filter at least one previously identified program from the load module execution information;

a correlator that correlates the filtered load module execution information with data that associates load module names with corresponding software products and develops a list of products executed in the computer over the course of a given time period; and

a reporter that outputs data reflecting the use of the software products in the computer in terms of software product names thereof.

16. (original) The system of claim 15, in which the monitor obtains the load module execution information from a load module table created by the operating system which makes

entries in the load module tables as processes are executed and access requests for load modules are made.

17. (original) The system of claim 15, in which the monitor is implemented to execute every time the end of a process is reached.

18. (original) The system of claim 15, in which the monitor executes as an exit routine near the end of a process.

19. (original) The system of claim 15, in which the monitor gathers and accumulates usage data across sub processes of a higher level process so that when the load module table is successively read, only those module entries not previously encountered in a prior sub process of the current high level process are accumulated and names of load modules already found in the table for the current high level process are ignored.

20. (original) The system of claim 15, in which the correlator operates by identifying the names of all software products used by correlating module usage data by using a knowledge base that associates the names of load modules with software products they comprise.

21. (original) The system of claim 15, in which the correlator operates by correlating module usage data with an inventory of software products that itself has been obtained by correlating in a knowledge base load module names with software product names.

22. (previously presented) A system for determining program usage on a computer, the system comprising:

a plurality of executable software programs constituting software products, each of the software products being constituted of one or more load modules, the load modules being stored in at least one memory of the computer;

an operating system of the computer that controls execution in the computer of software products through the invocation of respective load modules thereof;

a monitor that collects software product execution information by monitoring input or output to specific files or datasets by the software products; and

a filtering facility that is effective to filter at least one previously identified program from the software product execution information, wherein such inputs and outputs are associated with corresponding software products or groups of software products.

23. (original) The system of claim 22, in which the monitor is implemented as a background process.

24. (original) The system of claim 22, in which the monitor is implemented as an intercept systematically placed in either or both of a file open and/or file close system function of the computer.

25. (original) The system of claim 22, in which the monitor is operated as a batch process.

26. (previously presented) A system for determining non-usage of software products on a computer, the system comprising:

a plurality of executable software programs constituting software products, each of the software products being constituted of one or more load modules, the load modules being stored in at least one memory of the computer;

an operating system of the computer that controls execution in the computer of software products through the invocation of respective load modules thereof;

a software product surveyor that surveys the at least one memory of the computer and produces an inventory of the names of the load modules, the surveyor being operable with an associator that identifies and associates and records associations between product names and the load module inventory names;

a monitor that collects load module execution information over a given time period;

a filtering facility that is effective to filter at least one previously identified program from the load module execution information;

a correlator that correlates the filtered load module execution information with data that associates load module names with corresponding software products and develops a list of products executed in the computer over the course of a selected time period;

a comparing facility that compares information provided by the correlator to information provided directly or indirectly by the surveyor and which produces a list of unused software products; and

a reporter that outputs data reflecting the non-use of software products in the computer.

27. (original) The system of claim 26, in which the comparator is constructed to delete from the inventory those software product names which have been detected by the monitor as having been executed in the computer at least once during the given time period, leaving a list of non-used software products.

28. (previously presented) A system for determining program usage on a computer, the system comprising:

a plurality of executable software programs constituting software products, each of the software products being constituted of one or more load modules, the load modules being stored in at least one memory of the computer;

an operating system of the computer that controls execution in the computer of software products through the invocation of respective load modules thereof;

a monitor that collects load module execution information reflecting the usage of software products on the computer;

a filtering facility that is effective to filter at least one previously identified program from the load module execution information;

a library source determination facility that determines the load library from which each executed load module has been loaded; and

a reporter that outputs data showing respective directory paths for load modules that have been executed.

29. (original) The system of claim 28, in which the library source determination facility obtains a list of modules that have been used by a particular process, determines the load libraries and their search order used by the process, and using a search order determined in a prior step, searches the load libraries of the computer for a first library containing the same modules that best matches the list of modules used.

30. (original) The system of claim 28, in which the monitor operates as a concurrent process and module usage data and load library collection data are both obtained by the monitor and library usage is concurrently determined.

31. (original) The system of claim 29, in which the task of determining the correct load libraries in their appropriate search order is carried out as a separate process, wherein one of the module usage data or library collection data is obtained by the monitor and the other is obtained from a separate source and processed to determine load library usage.

32. (original) The system of claim 28, in which the library source determination facility uses a JCL (Job Control Language) interpreter.

33. (previously presented) The system of claim 28, in which the load library determination facility determines both the identity and order of load libraries used by a particular process by reading job control language ("JCL") data structures of a current job to obtain a load library list for the process.